PENDING CLAIMS:

1. (Previously Presented) For use in a processor, a branch architecture for limiting branch penalty without branch prediction comprising:

a fetch-branch unit operating in parallel with a decode unit and controlling retrieval of instructions for the decode unit, both the fetch-branch unit and the decode unit receiving the same instruction(s) during a given cycle, wherein the fetch-branch unit, upon detecting a branch instruction during one cycle,

initiates retrieval to both the fetch-branch unit and the decode unit of at least one sequential instruction from a location immediately following a location of a last retrieved instruction during one of a first cycle immediately following the one cycle and a second cycle immediately following the first cycle, and

initiates retrieval to both the fetch-branch unit and the decode unit of at least one target instruction from a target location for the branch instruction during the other of the first cycle immediately following the one cycle and the second cycle immediately following the first cycle.

2. (Previously Presented) The branch architecture as set forth in Claim 1 wherein the fetch-branch unit resolves the branch instruction and, upon resolving the branch instruction, causes both the fetch-branch unit and the decode unit to drop either the at least one sequential instruction or the at least one target instruction.

3. (Previously Presented) The branch architecture as set forth in Claim 2 wherein the fetch-branch unit, upon resolving the branch instruction, initiates retrieval to both the fetch-branch unit and the decode unit of at least one instruction from a location immediately following a location of a last retrieved instruction within either the at least one sequential instruction or the at least one target instruction, depending upon whether a branch is taken.

4. (Original) The branch architecture as set forth in Claim 1 wherein the fetch-branch unit, upon detecting a branch instruction during the one cycle, marks any fetched instruction preceding the branch instruction with a regular instruction type identifier, marks the branch instruction with a branch instruction type identifier, and marks any fetched instruction succeeding the branch instruction with a sequential instruction type identifier.

5. (Previously Presented) The branch architecture as set forth in Claim 4 wherein the fetch-branch unit, upon not detecting a branch instruction during the one cycle, marks all fetched instruction(s) with the regular instruction type identifier.

6. (Original) The branch architecture as set forth in Claim 1 wherein the fetch-branch unit marks the at least one sequential instruction with a sequential instruction type identifier.

7. (Original) The branch architecture as set forth in Claim 1 wherein the fetch-branch unit marks the at least one target instruction with a target instruction type identifier.

8. (Previously Presented) A processor comprising:

at least one execution unit;

a decode unit; and

a branch architecture for limiting branch penalty without branch prediction comprising:

a fetch-branch unit operating in parallel with the decode unit and controlling retrieval of instructions for the decode unit, both the fetch-branch unit and the decode unit receiving the same instruction(s) during a given cycle, wherein the fetch-branch unit, upon detecting a branch instruction during one cycle,

initiates retrieval to both the fetch-branch unit and the decode unit of at least one sequential instruction from a location immediately following a location of a last retrieved instruction during one of a first cycle immediately following the one cycle and a second cycle immediately following the first cycle, and

initiates retrieval to both the fetch-branch unit and the decode unit of at least one target instruction from a target location for the branch instruction during the other of the first cycle immediately following the one cycle and the second cycle immediately following the first cycle.

9. (Previously Presented) The processor as set forth in Claim 8 wherein the fetch-branch unit resolves the branch instruction and, upon resolving the branch instruction, causes both the fetch-branch unit and the decode unit to drop either the at least one sequential instruction or the at least one target instruction.

10. (Previously Presented) The processor as set forth in Claim 9 wherein the fetch-branch unit, upon resolving the branch instruction, initiates retrieval to both the fetch-branch unit and the decode unit of at least one instruction from a location immediately following a location of a last retrieved instruction within either the at least one sequential instruction or the at least one target instruction, depending upon whether a branch is taken.

11. (Original) The processor as set forth in Claim 9 wherein the fetch-branch unit, upon detecting a branch instruction during the one cycle, marks any fetched instruction preceding the branch instruction with a regular instruction type identifier, marks the branch instruction with a branch instruction type identifier, and marks any fetched instruction succeeding the branch instruction with a sequential instruction type identifier.

12. (Previously Presented) The processor as set forth in Claim 11 wherein the fetch-branch unit, upon not detecting a branch instruction during the one cycle, marks all fetched instruction(s) with the regular instruction type identifier.

13. (Original) The processor as set forth in Claim 8 wherein the fetch-branch unit marks the at least one sequential instruction with a sequential instruction type identifier.

14. (Original) The processor as set forth in Claim 8 wherein the fetch-branch unit marks the at least one target instruction with a target instruction type identifier.

15. (Previously Presented) For use in a processor, a method of processing branch instructions without branch prediction comprising:

operating a fetch-branch unit in parallel with a decode unit to control retrieval of instructions for the decode unit, wherein the same instruction(s) are retrieved to both the fetch-branch unit and the decode unit during a given cycle; and

upon detecting a branch instruction during one cycle,

initiating retrieval to both the fetch-branch unit and the decode unit of at least one sequential instruction from a location immediately following a location of a last retrieved instruction during one of a first cycle immediately following the one cycle and a second cycle immediately following the first cycle, and

initiating retrieval to both the fetch-branch unit and the decode unit of at least one target instruction from a target location for the branch instruction during the other of the first cycle immediately following the one cycle and the second cycle immediately following the first cycle.

16. (Previously Presented) The method as set forth in Claim 15 further comprising:

resolving the branch instruction; and

upon resolving the branch instruction, causing both the fetch-branch unit and the decode unit to drop either the at least one sequential instruction or the at least one target instruction.

17. (Previously Presented) The method as set forth in Claim 16 further comprising:

upon resolving the branch instruction, retrieving, to both the fetch-branch unit and the decode unit, at least one instruction from a location immediately following a location of a last retrieved instruction within either the at least one sequential instruction or the at least one target instruction, depending upon whether a branch is taken.

18. (Original) The method as set forth in Claim 15 further comprising:

upon detecting a branch instruction during the one cycle,

marking any fetched instruction preceding the branch instruction with a regular instruction type identifier,

marking the branch instruction with a branch instruction type identifier, and

marking any fetched instruction succeeding the branch instruction with a sequential instruction type identifier.

19. (Original) The method as set forth in Claim 18 further comprising:

upon not detecting a branch instruction during the one cycle, marking all fetched instruction with the regular instruction type identifier.

20. (Original) The method as set forth in Claim 15 further comprising:

marking the at least one sequential instruction with a sequential instruction type identifier; and

marking the at least one target instruction with a target instruction type identifier.